VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VYTVOŘENÍ INTERAKTIVNÍCH APLETŮ PRO PODPORU VÝUKY ZPRACOVÁNÍ OBRAZŮ

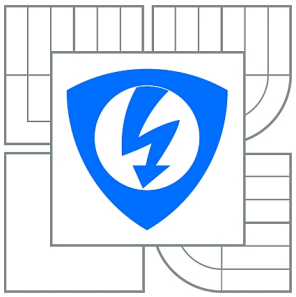
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JIŘÍ KŘIVÁNEK
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

VYTVOŘENÍ INTERAKTIVNÍCH APLETŮ PRO PODPORU VÝUKY ZPRACOVÁNÍ OBRAZŮ

INTERACTIVE APPLETS SUPPORTING TEACHING DIGITAL IMAGE PROCESSING

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

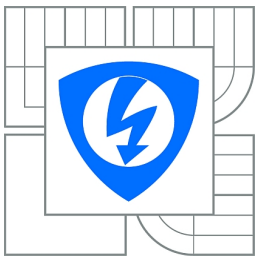
AUTOR PRÁCE — JIŘÍ KŘIVÁNEK

AUTHOR

VEDOUCÍ PRÁCE — Mgr. PAVEL RAJMIC, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Jiří Křivánek	***ID:***	110986
Ročník:	3	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Vytvoření interaktivních apletů pro podporu výuky zpracování obrazů

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a implementujte programy pro interaktivní podporu výuky v kurzech BZSG a MGMP, které budou mít podobu JAVA či Flash apletů. Témata apletů jsou: Konvoluce obrazů, Dithering obrazu, Bitové roviny obrazu, Gama korekce, Procedurální textury. Funkcionalita a ovládání vyplývají z diskuzí studenta s vedoucím práce.

DOPORUČENÁ LITERATURA:

[1] Žára, J. a kol.: Moderní počítačová grafika. Druhé vydání. Computer Press.

[2] Schildt, H.: Java7, výukový kurz. Computer Press, 2012.

[3] Aplety na stránce http://www.utko.feec.vutbr.cz/~rajmic/applets/

Termín zadání:	9.2.2015	***Termín odevzdání:***	2.6.2015

Vedoucí práce: Mgr. Pavel Rajmic, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se zabývá vytvořením aplikací pro podporu výuky počítačové grafiky. Cílem práce je vytvořit jednoduché aplikace, které budou názorně ukazovat jak jednotlivé metody fungují. Jsou zde implementovány metody konvoluce pro filtrování obrazů, dithering zabývající se distribucí chyby, bitové roviny, které zobrazují jednotlivé bitové roviny obrazu, gama korekce pro zesvětlení či ztmavení obrazu a procedurální textury. Všechny tyto metody jsou vytvořeny formou Java appletů.

KLÍČOVÁ SLOVA

Java, applet, konvoluce, dithering, Floyd-Steinberg, bitové roviny, procedurální textury, gama korekce

ABSTRACT

This thesis deals with the creation of applications for support of education computer graphics. The aim is to create a simple application which will show the functioning of the various methods. I implemented convolution methods for image filtering, dithering engaged in the distribution errors, bit planes that show each bit plane image, gamma correction to lighten or darken the image and procedural textures. All these methods are created in the form of Java applets.

KEYWORDS

Java, applet, convolution, dithering, Floyd-Steinberg, bit planes, procedural textures, gamma correction

KŘIVÁNEK, Jiří *Vytvoření interaktivních apletů pro podporu výuky zpracování obrazů*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2015. 40 s. Vedoucí práce byl Mgr. Pavel Rajmic, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Vytvoření interaktivních apletů pro podporu výuky zpracování obrazů“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Mgr. Pavlu Rajmicovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych chtěl poděkovat své rodině za podporu.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Při výuce v předmětech zabývající se počítačovou grafikou může být někdy složité pochopit jak určité metody na zpracování obrazů fungují, někteří studenti si někdy jen těžko dokáží představit pouze po přečtení textu jak určité kroky jednotlivých metod vlastně probíhají. Tato práce je proto zaměřena na tvorbu aplikací, které mají názorně ukazovat určité metody zpracování obrazu a podpořit tak výuku v těchto předmětech.

Pro tvorbu těchto aplikací byl vybrán jazyk Java a to ve formě Java appletů a to z důvodu, že applet běží v internetovém prohlížeči, který je už v dnešní době součástí snad každého počítače, nebo mobilního zařízení, takže v případě potřeby je možné ho kdykoliv rychle spustit bez nutnosti instalace nějakého dalšího softwaru.

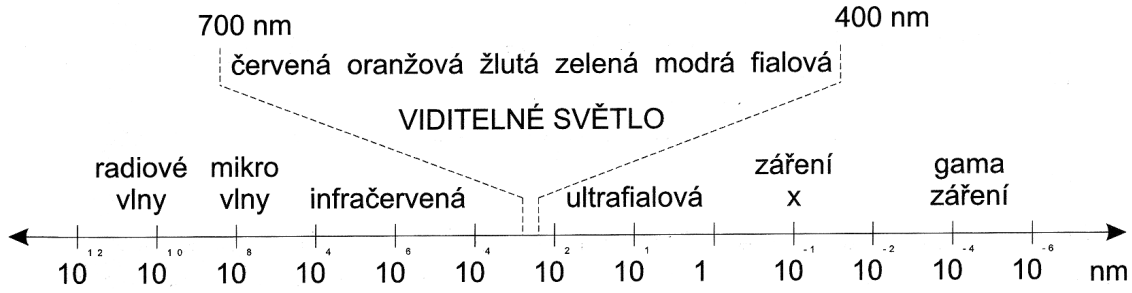
V této práci bude vysvětleno, navrženo a implementováno pět různých metod zpracování obrazu a to metody konvoluce obrazu, dithering, gama korekce, bitové roviny a procedurální textury.

V 1. části je vysvětleno co je to vlastně obraz a jak je interpretován, co je to barva a barevný prostor. Ve 2. části jsou teoreticky popsány jednotlivé metody, kterými se tato práce zabývá. Ve 3. části jsou vysvětleny funkce jednotlivých appletů a jejich ukázky.

1 POČÍTAČOVÁ GRAFIKA

1.1 Světlo a barvy

Člověk dokáže vnímat pouze určitou část elektromagnetického spektra, kterému se říká viditelné světlo. Elektromagnetické spektrum se skládá ze všech známých druhů záření, které je možno vidět na obr. 1.1. Elektromagnetické záření a jakákoliv látka mohou na sebe různým způsobem vzájemně působit a to v závislosti na vlnové délce. Pokud jsou obrazy pořízené při odlišných vlnových délkách, každý takový obraz může mít jiné vlastnosti a naprosto odlišně informovat o daných objektech či jevech. V oblasti viditelné části spektra (v rozmezí vlnových délek asi 380 – 720 nm) člověk vnímá záření s určitou vlnovou délkou jako barvu. Např. zelené světlo je dáno vlnovou délkou 550 nm a červené světlo o délce 720 nm.



Obr. 1.1: Spektrum elektromagnetického záření.[1]

Bílé světlo vznikne složením paprsků všech kmitočtů v daném pásmu, které vysílá určitý světelný zdroj. Pokud dopadne bílé světlo na daný objekt, některé kmitočty se na povrchu objektu odrazí a některé povrch pohltí. Barva objektu vznikne tedy kombinací všech kmitočtů přítomných v odraženém světle. Člověk bude vnímat objekt jako červený, budou-li v odraženém světle převládat nízké kmitočty. V tomto případě bude mít světlo dominantní kmitočet (dominantní vlnovou délku) na červeném konci spektra. Tento dominantní kmitočet je pojmenováván barvou, zabarvením světla či tónem.

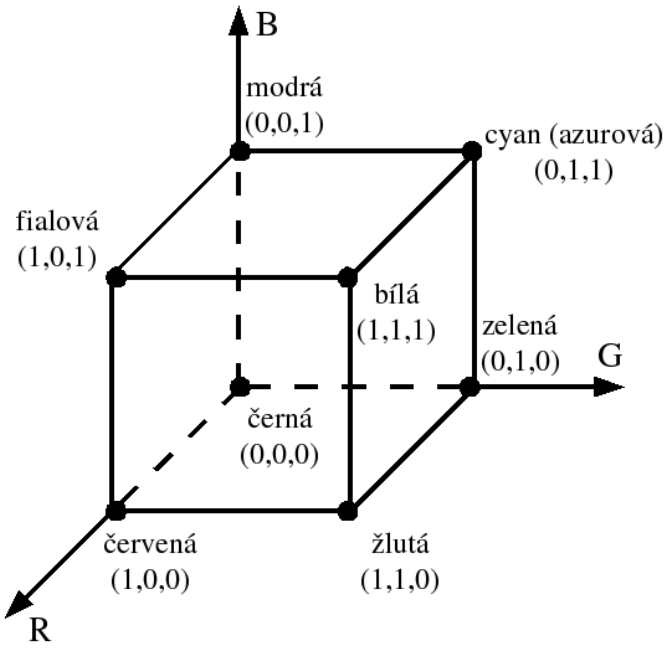
Kromě kmitočtu lze pro popis charakteristik světla použít i jiné vlastnosti. Lidské oko při pozorování zdroje světla je citlivé kromě barvy i na další podněty: jas, sytost a světlost.[2]

1.1.1 Barevné modely

Model RGB

Pro snímání barevného obrazu a zobrazení na monitoru se nejvíce využívá barevný model RGB. Barevný obraz podle tohoto modelu je složen aditivně z těchto tří barev: červené (red), zelené (green) a modré (blue). Každou barvu lze reprezentovat určitým počtem bitů. Obvykle se jedná o 8bit/barvu. Pro získání výsledné barvy stačí v určitém poměru složit jednotlivé barevné složky RGB. Tyto složky nabývají hodnot z intervalu $\langle 0, 1 \rangle$, nebo v celočíselném rozsahu $0-255$. Pokud vyjadřujeme barevné složky pomocí 3 bytů, každá barva má 8 bitů, tak lze formulovat 16 777 216 ($2^{3\times 8} = 2^{24}$) barev.

Jak už bylo zmíněno, barevný model RGB je označován za aditivní míchání barev. Na obr. 1.2 je vidět jednotková krychle, kde počátek souřadnic odpovídá černé barvě (R = 0, G = 0, B = 0) a vrchol o souřadnicích (R = 1, G = 1, B = 1) je roven bílé barvě.[4]



Obr. 1.2: Reprezentace barevného modelu RGB.[3]

Modely CMY a CMYK

Tento barevný model se používá pro tisk. Zkratka CMY vychází z těchto barev: Cyan (modrozelená), Magenta (purpurová) a Yellow (žlutá). Barevný model CMY je založen na subtraktivním míchání barev, kde počátek souřadnic odpovídá bílé barvě [0,0,0] a vrchol [1,1,1] černé, která vznikne právě subtraktivním složením předešlých

tří barev. Barvy CMY jsou doplňkové k modelu RGB. Pokud se základní barvy CMY překryjí, nevznikne čistá černá barva. Proto se zavedl i model CMYK, kde písmeno K značí černou (blacK = Key).[4]

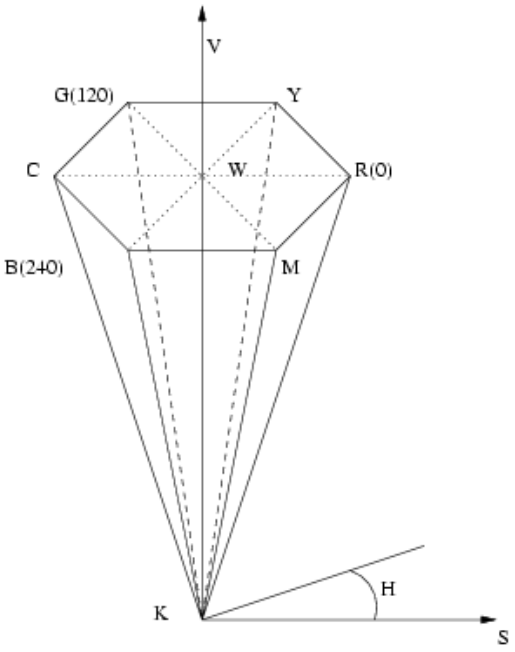
Vzájemný vztah pro převod mezi RGB a CMY:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{1.1}$$

Model HSV

Tento model popisuje barvy přirozenějším způsobem pro vnímání člověka, než předchozí uvedené modely. Model HSV nedefinuje barvu pomocí základních barev, ale díky třem hlavním parametrům jakými jsou:

- barevný tón (H, hue) – určuje polohu barevného odstínu na tzv. barevném kole (0° až 360°). Každá základní barva má svůj daný úhel (360° = červená, 120° = zelená a 240° = modrá).
- sytost (S, saturation) – udává množství šedi v poměru k odstínu barvy, kterou docílíme díky příměsi bílé barvy. Měří se v procentech od 0 % (šedá) do 100 % (plně sytá barva).
- jasová hodnota (V, value) – jedná se o světlost nebo tmavost barvy a tedy množství bílého světla. Vyjadřuje kolik světla dokáže barva odrazit.[5]



Obr. 1.3: Geometrická reprezentace modelu HSV.[6]

Nevýhodou tohoto modelu je, že přechod mezi černou a bílou není plynulý a změna barevného tónu neprobíhá po kružnici, ale po šestiúhelníku a změna tónu tak rovněž není plynulá.

Model YCbCr

Jedná se o subtraktivní model stejně jako CMY. Jas je roven složce Y a složky Cb a Cr jsou modrý a červený chromatický signál. Tento model se používá při zápisu rastrových obrázků ve formátu JPEG a u digitálních fotoaparátů. Model využívá toho, že lidské oko dokáže velmi dobře vnímat změnu světelnosti, ale nedokáže zachytit malé změny chromatických signálů. Při ztrátě 50% informací v chromatických kanálech jsou změny téměř nepostřehnutelné.[5]

Převod z modelu RGB do YCbCr:

$$Y = 0{,}31R + 0{,}59G + 0{,}11B, \tag{1.2}$$

$$Cr = 0{,}713(R - Y), \tag{1.3}$$

$$Cb = 0{,}564(B - Y). \tag{1.4}$$

Šedotónový obraz

Každý pixel v šedotónovém obraze udává informaci o intenzitě daného bodu. Tato hodnota vyjadřuje odstín šedé barvy v rozmezí od 0 do 255, kde 0 udává nejtmavší odstín, což je černá barva a hodnota 255 udává nejsvětlejší odstín, což je bílá barva. Při převodu barevného obrazu, který má tři barevné složky R,G,B na šedotónový obraz se z jednotlivých barevných složek vypočítá intenzita výsledného bodu podle vzorce

$$I = 0{,}299R + 0{,}587G + 0{,}114B. \tag{1.5}$$

Tento vzorec je dán citlivostí lidského oka na jednotlivé barevné složky, kde nejcitlivěji vnímá změnu zelené složky a nejméně vnímá změnu modré složky.

U modelu RGB platí, že pokud mají všechny tři barevné složky stejnou hodnotu, výsledná barva udává právě jeden odstín šedi. Jestliže tedy chceme uložit obraz v modelu RGB jako šedotónový, přiřadíme všem třem složkám hodnotu intenzity vypočítané podle předchozího vzorce. Hodnoty jednotlivých složek tedy budou $I = R = G = B$.

1.2 Obraz a jeho reprezentace

Obraz je chápán jako vícerozměrný signál a můžeme z něj zjistit velikost, polohu, průměrný jas a další vlastnosti určitého objektu. Bývá charakterizován pomocí matematické spojité skalární funkce f dvou nebo tří proměnných a je nazývána obrazovou funkcí. Obrazová funkce o dvou souřadnicích $f(x, y)$ popisuje statický obraz a funkce o třech proměnných $f(x, y, t)$ nebo $f(x, y, z)$ se použije v případě plošných obrazů, které se mění v čase nebo v případě objemových obrazů.

Pro zpracování v počítači je potřeba nejdříve obrazovou funkci digitalizovat. K tomuto ději se používá vzorkování obrazu v matici $M \times N$ bodů a kvantování do K intervalů. Obě dvě metody vytváří ze spojitého signálu signál diskrétní, což má za následek ztrátu informace. Tato ztráta se dá úplně minimalizovat, když je vzorkovací frekvence dvakrát větší, než největší frekvence obsažená v původním signálu.

- Vzorkování – časovou osu rozdělíme na stejné intervaly a v každém tomto intervalu vezmeme jeden vzorek původního signálu.
- Kvantování – vzorky získané vzorkováním jsou taktéž spojité a kvantováním se z nich stanou vzorky diskrétní, tak že se každý vzorek přiřadí do určité kvantovací hladiny, které jsou předem určeny. Tato operace je ztrátová a velikost chyby záleží na počtu kvantovacích hladin. Čím více je použito kvantovacích hladin, tím je chyba menší.

Obrazová funkce $f(x, y)$ má zde podobu matice. Tato matice je tvořena nejmenšími, dále nedělitelnými obrazovými elementy, nebo-li pixely. Název vychází z anglického spojení picture element. Nulové souřadnice bodů x, y se nachází v levém horním rohu matice. V každém pixelu se nachází informace o jeho barvě. Počet bitů a formát ve kterém je informace uložena záleží na použitém barevném modelu ve kterém je obraz uložen. Počet bitů v pixelu také udává tzv. barevnou hloubku.[7]

Používané barevné hloubky:

- 1bitová barva ($2^1 = 2$ *barvy*) také označováno jako Mono Color
- 4bitová barva ($2^4 = 16$ *barev*)
- 8bitová barva ($2^8 = 256$ *barev*)
- 15bitová barva ($2^{15} = 32\ 768$ *barev*) také označováno jako Low Color
- 16bitová barva ($2^{16} = 65\ 536$ *barev*) také označováno jako High Color
- 24bitová barva ($2^{24} = 16\ 777\ 216$ *barev*) také označováno jako True Color
- 32bitová barva ($2^{32} = 4\ 294\ 967\ 296$ *barev*) také označováno jako Super True Color
- 48bitová barva ($2^{48} = 281\ 474\ 976\ 710\ 656$ *barev*) také označováno jako Deep Color

2 METODY PRO ZPRACOVÁNÍ OBRAZU

2.1 Konvoluce obrazu

Zjednodušeně lze říci, že konvoluce je matematický operátor zpracovávající dvě funkce. Používá se pro filtraci obrazu a hlavně k vyhlazování obrazu a zvýrazňování hran. U spojité konvoluce se jedná o jednorozměrné funkce $f(x)$ a $g(x)$. V případě masky obsahující 3 prvky konvoluci vyjádříme

$$g(x) = \sum_{s=-1}^{1} h(s)f(x+s) = h(x-1)f(x-1) + h(x)f(x) + h(x+1)f(x+1). \quad (2.1)$$

2.1.1 Diskrétní dvourozměrná konvoluce

U diskrétní dvourozměrné konvoluce se tentokrát pracuje s funkcemi $f(x, y)$ a $h(x, y)$. Funkce $h(x, y)$ je tzv. maska, případně konvoluční jádro (kernel). Konvoluce se dá rozepsat do tří kroků. Prvním krokem je obrácení masky h o 180°, dále posun masky h vzhledem k funkci f pomocí změny (x, y) a nakonec vypočtení sumy součinů veškerých koeficientů masky h pro každé posunutí (x, y). Pokud uvážíme masku 3×3 postup zapíšeme následovně

$$g(x, y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} h(s, t)f(x - s, y - t). \quad (2.2)$$

Konvoluci lze zapsat

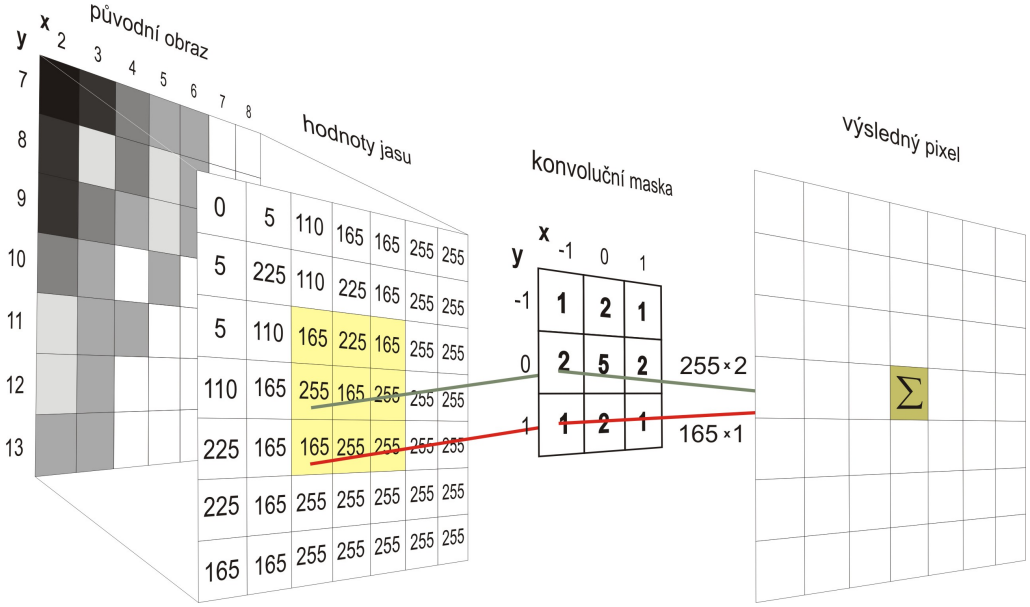
$$g(x, y) = h * f = \sum_{s=-s_{max}}^{s_{max}} \sum_{t=-t_{max}}^{t_{max}} h(s, t)f(x - s, y - t). \quad (2.3)$$

Existuje široká škála konvolučních masek, kde každá maska slouží pro jiný účel ve zpracování obrazu. Díky těmto maskám dosáhneme spousty operací a nadefinujeme různé filtry (dolní propust, horní propust či gradientní operátory).[4]

2.2 Vyhlazování obrazu

Obraz může být ovlivněný nežádoucími jevy, jako je např. šum, který se projevuje jako zrnění. A právě vyhlazení obrazu slouží k potlačení těchto nežádoucích artefaktů. Metody založené na vyhlazování obrazu lze rozdělit na lineární a nelineární.

Lineární metody pracují na principu diskrétní konvoluce popsané v kapitole 2.1.1. Tato metoda je charakteristická tím, že novou hodnotu daného pixelu vypočítá jako lineární kombinaci hodnot vybraného okolí. Lineární filtry se od sebe liší pouze v použití jiné konvoluční masky $h(x, y)$.



Obr. 2.1: Princip dvourozměrné konvoluce. [8]

2.2.1 Vyhlazování průměrováním

Jedna z metod vyhlazování obrazu je obyčejné průměrování. Je zde použita maska, pomocí které je výsledek konvoluce průměr hodnot jasu z okolí bodu v obraze:

$$g(x,y) = \frac{1}{M} \sum_{s=-s_{max}}^{s_{max}} \sum_{t=-t_{max}}^{t_{max}} f(x-s, y-t), \tag{2.4}$$

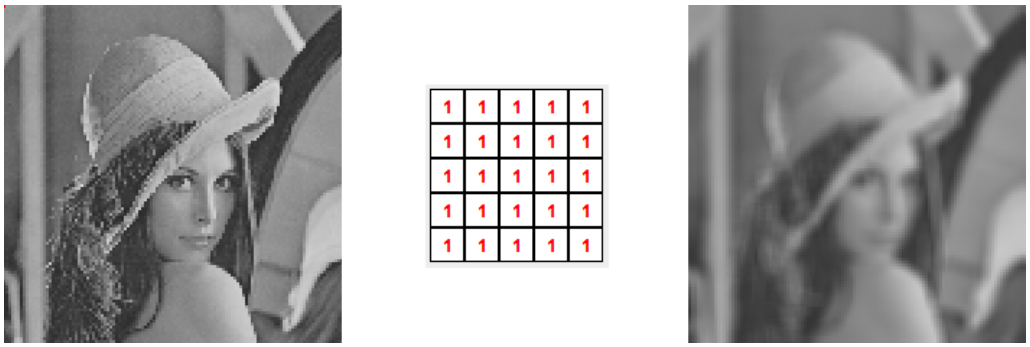
kde M je počet bodů v masce, $M = (2s_{max}+1)\cdot(2t_{max}+1)$. Pro velikost okolí 3×3 vyjde:

$$g(x,y) = \frac{1}{9} \sum_{s=-1}^{1} \sum_{t=-1}^{1} f(x-s, y-t) \tag{2.5}$$

a konvoluční maska vypadá následovně:

$$h = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \tag{2.6}$$

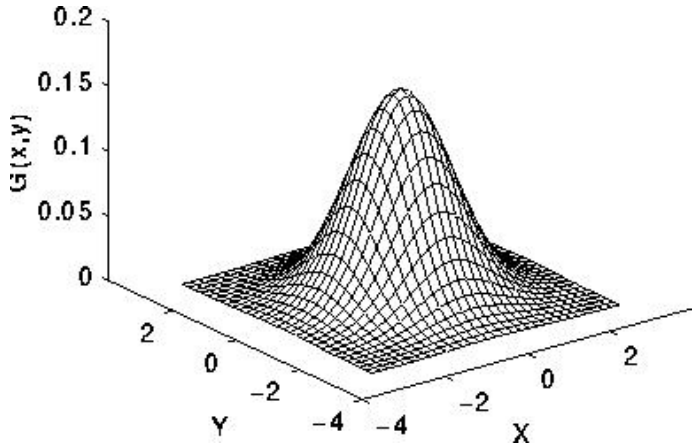
Ovšem při použití metody obyčejného průměrování se mohou rozmazávat hrany v obraze, což je nevýhodou. Proto je průměrování pomocnou metodou pro výpočet střední hodnoty jasu a s tímto výsledkem dále pracují nelineární metody vyhlazování. [4]



Obr. 2.2: Ukázka filtru jednotková matice.

2.2.2 Gaussovo vyhlazování

U metody zvané Gaussovo vyhlazování mají koeficienty blíže středu masky vyšší váhu a odpovídají hodnotám na Gaussově křivce.



Obr. 2.3: Ukázka Gaussova rozdělení.[9]

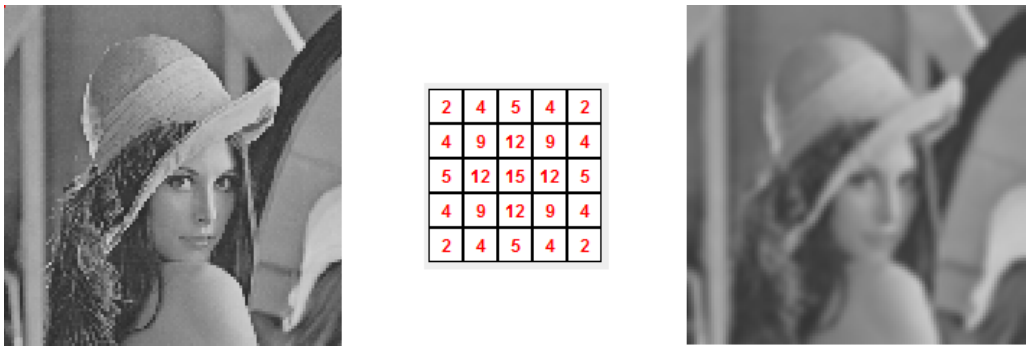
Na obr. 2.3 je znázorněn tvar hustoty pravděpodobností $g(x, y)$ pro dvourozměrný náhodný vektor s Gaussovým rozdělením. Dvourozměrné Gaussovo rozdělení se střední hodnotou (0,0) je definované vztahem:

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2.7}$$

kde σ^2 je rozptyl. Strmost Gaussové funkce a velikost masky závisí na parametru σ.[4]

2.2.3 Mediánová filtrace

Nelineární vyhlazovací metody dokáží částečně potlačit rozmazané hrany v obraze. Ve zkoumaném okolí se snaží najít část, do které patří reprezentativní bod. Pouze



Obr. 2.4: Ukázka filtru Gaussovo rozostření.

pixely, které jsou v této oblasti jsou použity pro hledání jasové hodnoty, která bude reprezentovat celé okolí ve výstupním obraze.

Známá nelineární metoda je mediánová filtrace. Zde se posouvá pomyslná maska po obraze a vybírá se medián z hodnot ležících pod touto maskou. Nejde o konvoluci, protože se jedná o statický filtr. Při hledání mediánu je nejprve potřeba vstupní hodnoty pixelů uspořádat podle velikosti a to vzestupně. Medián je roven hodnotě prvku s pořadovým číslem nejbližším jedné polovině počtu vstupních prvků. Mediánová filtrace vykazuje přijatelné vlastnosti při potlačení šumu typu „pepř a sůl“. [4]

2.2.4 Filtry ve frekvenční oblasti

Filtrace obrazu ve frekvenční oblasti je založená na principu, že každou funkci $f(x)$ lze rozložit na součet sinusových a kosinusových harmonických funkcí odlišných frekvencí, které jsou následně vynásobeny váhovým koeficientem nebo váhovou funkcí. Pokud se jedná o periodickou funkci, hovoříme o Fourierově řadě a u neperiodické funkci lze hovořit o Fourierově transformaci.

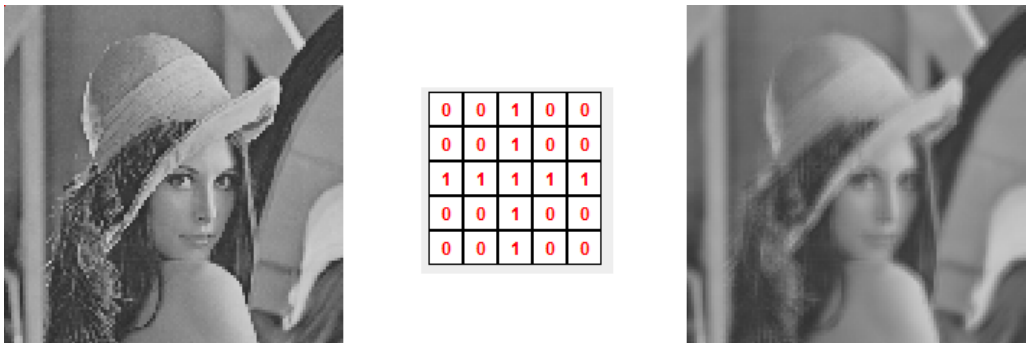
Dolní propust

Ideální filtr typu dolní propusti ořízne všechny harmonické složky spektra, u kterých je vzdálenost od počátku spektra větší než definovaná hranice D_0.

Přenosová funkce ideálního filtru dolní propusti je

$$H(u,v) = \begin{cases} 1 & \text{když } D(u,v) \leq D_0 \\ 0 & \text{když } D(u,v) > D_0 \end{cases}, \tag{2.8}$$

kde $D_0 > 0$ a $D(u,v)$ představují vzdálenost bodu (u,v) od počátku filtrační funkce $H(u,v)$.



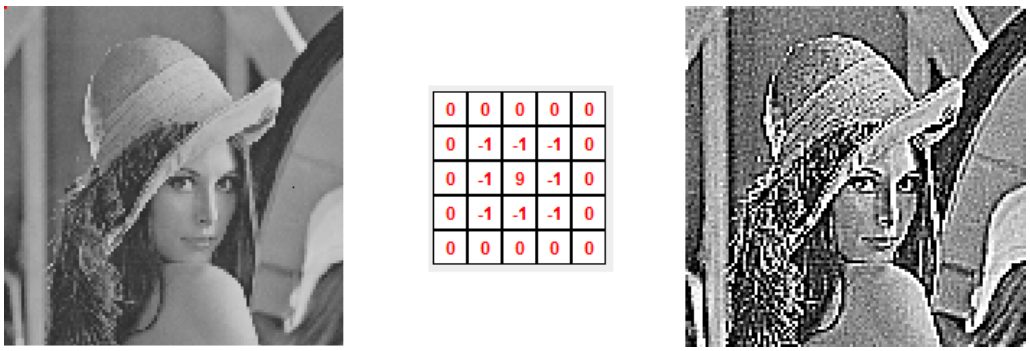
Obr. 2.5: Ukázka filtru dolní propust.

Horní propust

Na rozdíl od dolní propusti, kde se ořezávají vysokofrekvenční složky, zde se oříznou nízkofrekvenční složky spektra, u kterých je vzdálenost od počátku centrovaného spektra menší než definovaná hranice D_0.[10]

Přenosová funkce ideálního filtru horní propusti má tvar

$$H(u,v) = \begin{cases} 0 & \text{když } D(u,v) \leq D_0 \\ 1 & \text{když } D(u,v) > D_0 \end{cases} . \tag{2.9}$$



Obr. 2.6: Ukázka filtru horní propust.

2.3 Dithering obrazu

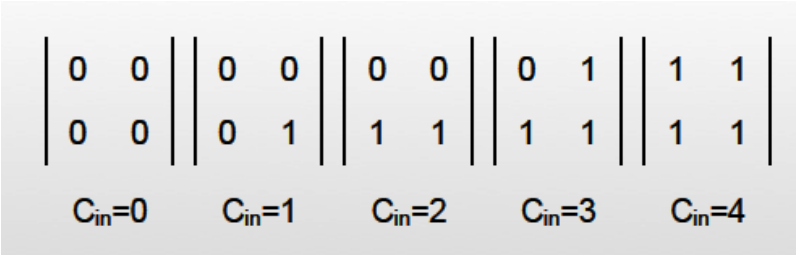
Při redukci počtu barev v obraze dochází ke ztrátě informací, proto existují metody, které dokáží tuto ztrátu minimalizovat. Metoda ditheringu (jinými slovy rozptyl) spočívá ve vytváření polotónového obrazu tak, aby zůstala zachovaná původní vizuální informace a to v co největší míře. Každý pixel původního obrazu je nahrazený novou hodnotou podle vybrané metody. Následující metody budou popisovat převod šedotónového obrazu na černobílý.

2.3.1 Náhodný rozptyl

Pomocí generátoru náhodných čísel je vygenerováno náhodné číslo, které udává práh, podle kterého se rozhoduje, zda bude pixel černý nebo bílý. Pokud je hodnota vstupního pixelu menší než práh, výsledný pixel bude černý a pokud je větší, výsledný pixel bude bílý. Tato metoda nevykazuje úplně nejlepší výsledky.

2.3.2 Maticový rozptyl

Tato metoda nahrazuje vstupní hodnotu odstínu šedi maticí složenou pouze z černých a bílých bodů. Velikost matice je dána podle rozsahu vstupních intenzit. Například pro rozsah vstupních hodnot $\langle 0, 4\rangle$ bude vytvořeno 5 vzorů podle obr. 2.7. Jsou dvě možnosti jak k této metodě přistupovat, buď se vezme pouze jeden vstupní pixel a ten se nahradí odpovídající maticí, což znamená, že se obraz zvětší nebo se vezme matice pixelů stejně velká jako jsou definované vzory, všechny vstupní pixely se zprůměrují a podle výsledku se vybere jeden ze vzorů, který se vloží na místo původních pixelů.



Obr. 2.7: Maticové vzory pro 5 úrovní intenzity.

2.3.3 Distribuce chyby – Floyd-Steinberg

Zpracovávání pixelů probíhá po řádcích od shora dolů a začíná se zleva doprava. Při dokončení řádku se neskáče na začátek dalšího, ale další řádek se čte zprava doleva.

Aktuální pixel je zpracován tak, že k jeho vstupní intenzitě se najde nejbližší barva v paletě, tato barva je pak novou barvou právě zpracovávaného pixelu. Od původní barvy se odečte nová barva a vyjde chyba, která se distribuuje na následující okolní pixely tedy ty, které ještě nebyly zpracovány. Chyba se rozdělí mezi pixely podle určitých poměrů, které jsou zobrazeny na obr. 2.8. Tyto poměry jsou ovšem neceločíselné a tato metoda pracuje s celými čísly, proto všechny podíly chyby zaokrouhlíme na celá čísla až na poslední, ten určíme tak že od celkové chyby odečteme součet všech předchozích podílů a zbytek přičteme k poslednímu pixelu. Tato metoda prokazuje velmi dobré výsledky.[2]

0	0	0
0	X	7/16
3/16	5/16	1/16

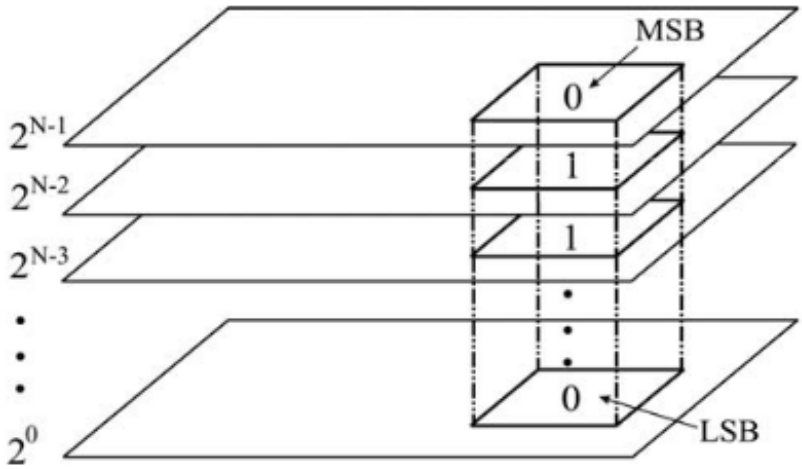
Obr. 2.8: Distribuce chyby podle Floyd-Steinberg.

2.4 Bitové roviny obrazu

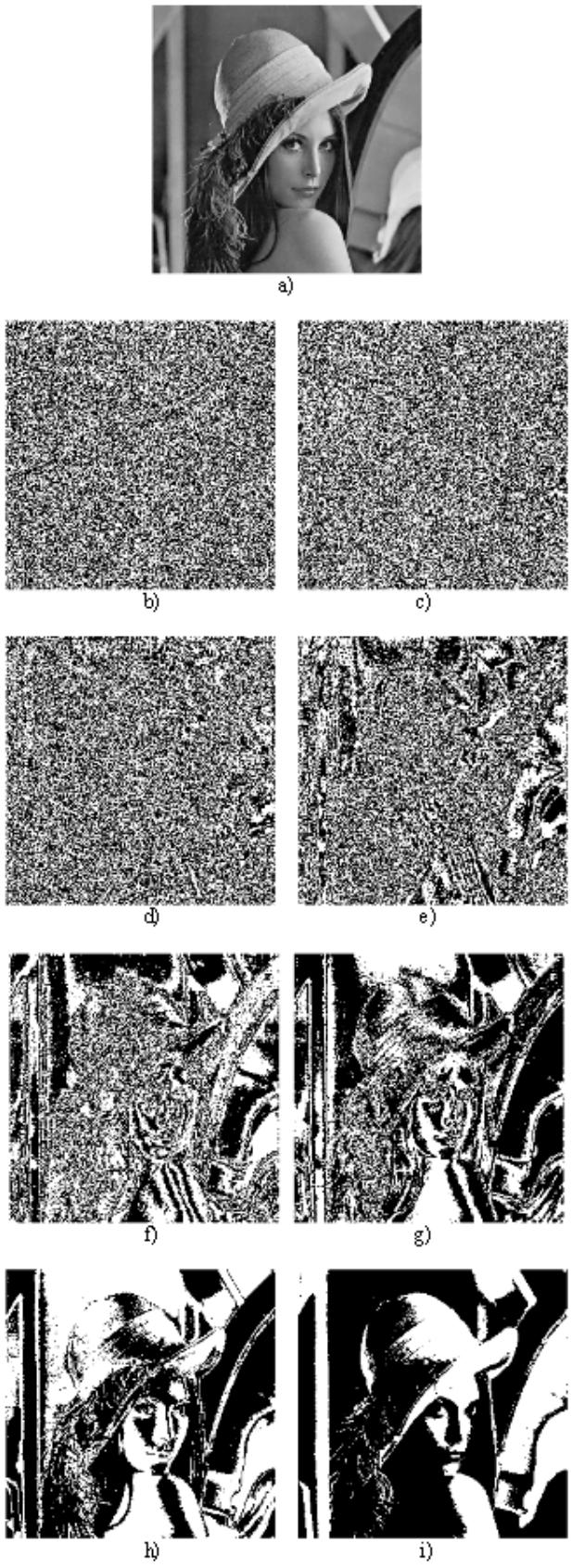
Každý pixel šedotónového obrazu obsahuje hodnotu intenzity, která nabývá hodnot $\langle 0, 255 \rangle$, tzn. že každý pixel má v sobě uloženo osmibitové číslo. Každý bit má svoji váhu podle rovnice

$$px = p_0 2^0 + p_1 2^1 + p_2 2^2 + p_3 2^3 + \cdots + p_{n-1} 2^{n-1}. \tag{2.10}$$

Když se vezme z každého pixelu jeden byt stejné váhy a zobrazí se jako černobílý obraz, zobrazí se právě jedna bitová rovina obrazu. Nejnižší bitová rovina je s váhou 0, ta je nejméně významná a má nejmenší vliv na výsledný obraz. Naopak bitová rovina s váhou 7 je rovina s největším vlivem na výsledný obraz.[2]



Obr. 2.9: Rozklad obrazu na bitové roviny.[11]



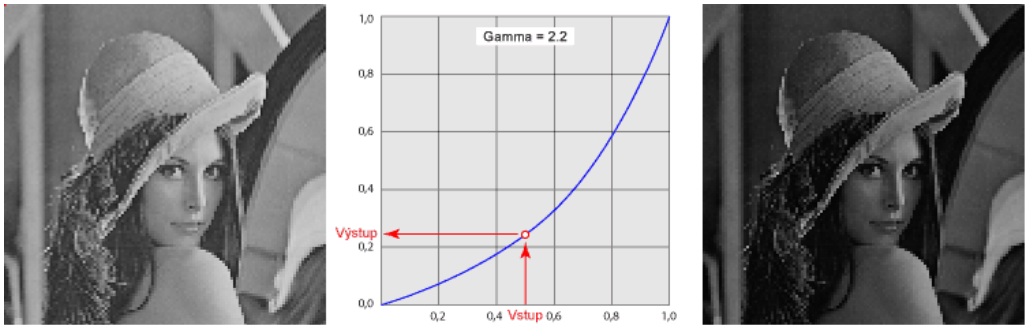
Obr. 2.10: Rozklad obrazu na bitové roviny.

2.5 Gama korekce

Gama korekce je vlastně zesvětlení či ztmavení obrazu. Tato metoda je užitečná především u monitorů, které na vstupní jas reagují nelineárně a pixel o intenzitě 0,8 zobrazí s intenzitou pouze 0,3. Tato nelinearita je vyvážena podle vzorce

$$I_{vst} = I_{vys}^{\gamma}. \tag{2.11}$$

Při úpravě světlosti fotografie má gama korekce výhodu v tom, že nemění okrajové body, tj. černá a bílá zůstávají pořád stejné a podle gama se mění jen odstíny šedé.[2]



Obr. 2.11: Ukázka křivky a změny jasu obrazu pro gama=2,2.

U barevných obrazů má gama vliv také na barvu. Pokud se obraz zesvětlí pomocí gama, přidá se do všech barev bílá a sníží se tak sytost barvy. Pokud se obraz ztmaví, od všech barev se bílá odečte a sytost se tak zvýší.

2.6 Textury

Textura je vzorek, který se nanáší na povrch objektu a udává tak jeho vzhled. Základní jednotkou textury je *texel*. Vlastnosti objektu, které textura udává je barevnost, svítivost, průhlednost, odraz světla a hrbolatost. Podle rozměrů se textury dělí na jedno, dvou, troj a čtyřrozměrné. Další významné dělení je na rastrové a procedurální textury.

- Rastrové textury – jsou ve formě obrazového souboru, jejich nevýhodou je, že zabírají velký prostor v paměti.
- Procedurální textury – jsou definovány na základě procedury a tím nejsou tak náchylné na paměťový prostor. Další výhodou je, že je možné měnit jejich vzhled jen změnou určitých parametrů v proceduře.

2.6.1 Procedurální textury

Procedurální textury jsou generované pomocí matematických vzorců. Používají se především pro textury, které realizují určité opakující se vzory nebo textury představující šum.

Perlinova šumová funkce

Perlinova funkce je rychlá, invariantní, spojitá, má omezené frekvenční spektrum a je opakovatelná. Základem je šumová funkce $noise(float\ x, float\ y, float\ z)$, která pro hodnoty $[x, y, z]$ vrací vždy stejné náhodné číslo z intervalu $\langle$-1, 1$\rangle$. Hlavní myšlenkou je generování spojitého šumu v diskrétním prostoru. Tento prostor je rozdělen do pravidelné mřížky, jejíchž vrcholy se značí $[i, j, k]$ a v každém z nich je definovaná funkce, které se říká *vlnka*.

Výpočet Perlinovy funkce pro bod $[x, y, z]$ se skládá ze tří kroků. První krok je určení buňky a to zaokrouhlením reálných hodnot $[x, y, z]$ na nejbližší dolní celočíselnou hodnotu a tím se určí souřadnice levého dolního rohu krychle. Druhým krokem je výpočet tvaru vlnky, která má v bodě $[i, j, k]$ nulovou hodnotu a pro určení tvaru stačí hodnota jejího gradientu pro střed vlnky. Pro výpočet se používá pole pseudonáhodných vektorů G o velikosti 256. Přístup do pole G se provádí přes pole P, které obsahuje náhodné permutace indexů. Toto pole je však jednorozměrné a souřadnice $[i, j, k]$ trojrozměrné, proto se provádí přeložení souřadnic pomocí funkce

$$fold(i, j, k) = P[(P[(P[i\ mod\ 256] + j)\ mod\ 256] + k)\ mod\ 256]. \tag{2.12}$$

Dále se vypočítá relativní vzdálenost souřadnic $[x, y, z]$ od $[i, j, k]$

$$[u, v, w] = [x, y, z] - [i, j, k]. \tag{2.13}$$

Pokles hodnoty funkce se vzdáleností je dán funkcí

$$drop(t) = 1 - 3|t|^2 + |t|^3. \tag{2.14}$$

Celkový úbytek v bodě $[u, v, w]$

$$\Omega(u, v, w) = drop(u) * drop(v) * drop(w). \tag{2.15}$$

Hodnota vlnky v bodě $[i, j, k]$ je určena:

$$\Omega(u, v, w) * G(i, j, k). \tag{2.16}$$

Třetí krok je určení hodnoty funkce v bodě $[x, y, z]$, která se vypočítá sečtením hodnot vlnek všech vrcholů krychle.

Skládání šumových funkcí

Složení šumových funkcí je součet funkcí $noise(x, y, z)$, každá se změněnou amplitudou a frekvencí. Jednotlivým složkám se říká oktávy. Součet probíhá podle vzorce

$$snoise(x, y, z, p, n) = \sum_{i=0}^{n-1} a_i * noise(f_i x, f_i y, f_i z), \tag{2.17}$$

kde n je počet oktáv, p patřící do intervalu $(0, 1)$ je persistence, která určuje rychlost klesání vlivu oktávy. To zajišťuje amplituda a_i i-té oktávy $a_i = p^i$. Frekvence se vypočítá $f_i = 2^i$.

Součet šumových funkcí se uplatňuje především při výpočtu modelů krajin, součet absolutních hodnot pak pro modely podobné mrakům. Textury mramoru nebo dřeva se vypočítá pomocí barevné rampy, která je modulována Perlinovou funkcí. Výpočet mramoru vypadá následovně

$$rampa(x) = 1/2(1 + \sin(x)) \tag{2.18}$$

$$mramor(x, y, z) = rampa(x + c * snoise(x, y, z, p, a, n)), \tag{2.19}$$

kde c je tzv. turbulence, která udává deformaci textury.[2]

3 NÁVRH APLIKACE

Všechny aplikace jsou implementovány v jazyce Java a to ve formě Java Appletů. Pro jejich tvorbu jsem používal vývojové prostředí Eclipse. Pro vytvoření grafů jsem použil balíček JFreeChart. V této části si detailněji představíme jednotlivé applety všech metod zpracování obrazu, kterými se tato práce zabývá. Applety mají kompletně navržené uživatelské rozhraní, které bude ukázáno a také jsou kompletně implementovány a bude zde tak popsána i jejich funkčnost.

Grafické rozhraní appletů je vytvořeno na základě knihovny *javax.swing*. Rozložení všech appletů vychází z jednoho základního rozložení komponent, ve kterém jsou dominantní dva panely pro vykreslování obrazů. Vlevo je vstupní obraz a vpravo výstupní obraz po použití dané metody. Mezi těmito obrazy je střední ovládací panel, ve kterém může uživatel nastavit různé parametry programu. V dolní části okna se pak názorně ukazují jednotlivé operace dané metody.

3.1 Java applet

Applet je program, který je natažen do internetového prohlížeče a nikde jinde než v něm nemůže být spuštěn. Applet běží v prostředí *Java virtual machine*, které je nainstalováno s prohlížečem nebo si musí uživatel nainstalovat sám určitý plugin. Takovýto běh programu způsobuje určitá bezpečnostní rizika, která jsou však zmírněna tím, že applet nemůže přistupovat k lokálnímu souborovému systému a nemůže využívat externí knihovny, toto se dá ovšem určitými metodami obejít.

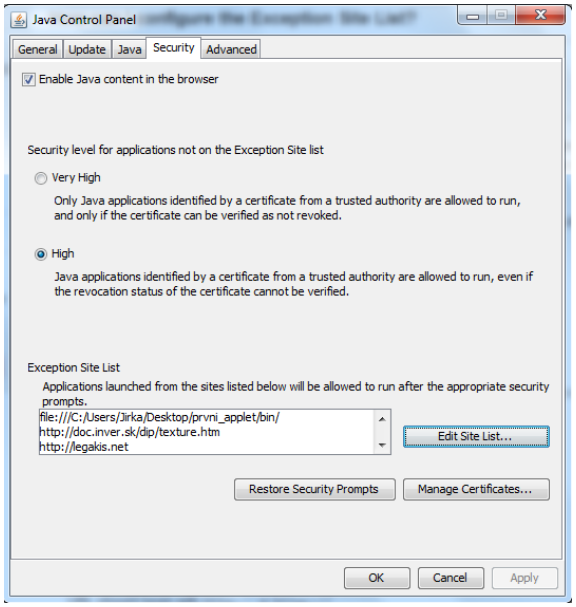
Applet se dále liší od klasického programu v jazyce Java tím, že nemá hlavní funkci `main()`, ale hlavní třída je podtřídou třídy *Applet* nebo *JApplet* a má životní cyklus, který zajišťují funkce `init()`, `start()`, `stop()`, `destroy()`. Do prohlížeče je applet vložen pomocí značky `<APPLET>`.

3.2 Bezpečnost Java appletů

Java vzhledem ke svému velkému rozšíření je jedním z nejvíce napadaných softwarů a často obsahuje velké zabezpečovací chyby, které vedou k nepříliš dobré pověsti Javy. Proto firma přišla s takovým řešením, že každý applet musí být podepsán certifikovaným podpisem od jedné z certifikačních autorit. V případě že si je však uživatel jistý že daný applet není nebezpečný může toto řešení obejít a to tak, že adresu příslušného appletu vloží do seznamu výjimek tzv. *Exception Site List*. Poté už nebude vyžadován podpis a uživateli se pouze zobrazí upozornění, že applet není podepsán a to potvrdí tlačítkem *Run*.

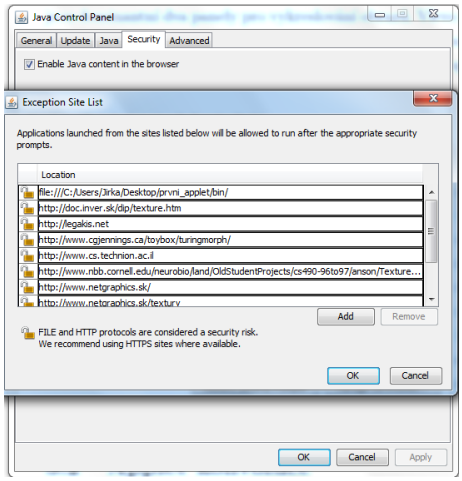
Postup vložení stránky do Exception site listu:

- Otevřít Java Control Panel a kliknout na záložku Security



Obr. 3.1: Java Control Panel.

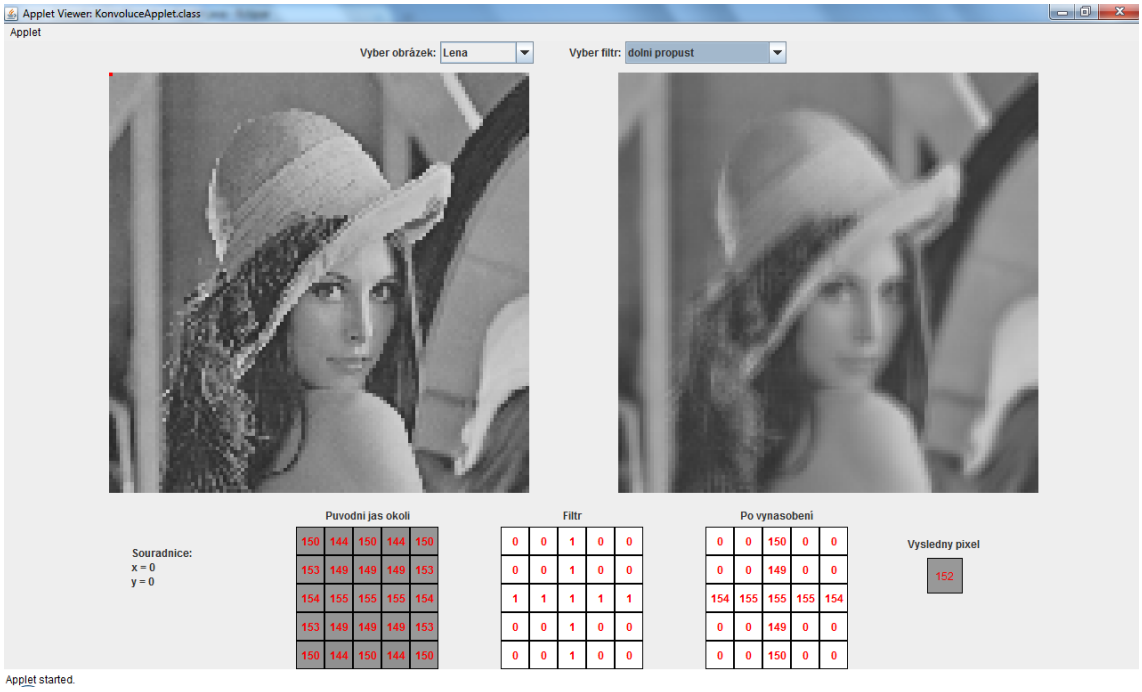
- Kliknout na tlačítko Edit Site List...
- Po otevření Exception Site Listu vložit tlačítkem *Add* stránku s příslušným appletem. Stránka by měla začínat *http://*.



Obr. 3.2: Exception Site List.

3.3 Applet konvoluce

Applet konvoluce ukazuje názorně jak probíhá tato metoda v jednotlivých krocích. Rozvržení uživatelského rozhraní je zobrazeno na obr. 3.3. Obraz vlevo je originální vstupní obraz, který je možno vybrat ze vstupní sady obrazů, pomocí rolovací nabídky nahoře nad obrazem. Obraz vpravo je výstupní obraz po konvoluci, který byl filtrován pomocí vybraného filtru, které je možné také vybírat v horním panelu. V dolní části okna pod obrazy je znázorněn průběh zpracování aktuálního pixelu. Vlevo jsou zobrazeny jeho souřadnice ve vstupním obraze. Následují tři matice, z nichž první ukazuje barvu a hodnotu aktuálního pixelu a jeho okolí. Druhá matice je použitý filtr a třetí matice zobrazuje hodnoty po vynásobení jednotlivých prvků předchozích matic. Úplně vpravo je pak zobrazen výstupní pixel, který vznikne sečtením všech prvků třetí matice a vydělením tohoto součtu váhou, která je dána použitým filtrem.



Obr. 3.3: Uživatelské rozhraní Appletu konvoluce.

3.3.1 Funkčnost appletu

Při spuštění appletu se okamžitě načte výchozí vstupní obraz, který je automaticky filtrován výchozím filtrem, kterým je dolní propust a výsledný obraz je zobrazen na pravé straně. Jako aktuální pixel je vybrán pixel o souřadnicích $x = 0$, $y = 0$

a je označen ve vstupním obraze červenou barvou. V dolním panelu jsou zobrazeny hodnoty tohoto pixelu a hodnoty všech operací, které s ním byly provedeny.

Uživatel má poté možnost zkoumat všechny pixely vstupního obrazu a to kliknutím na požadovaný pixel myší. Poté se zobrazí jeho souřadnice a všechny hodnoty s ním spojené. Dále má také možnost v horním panelu vybrat jiný obraz nebo jiný filtr. Ihned po výběru jednoho nebo druhého se konvoluce znovu provede.

3.3.2 Implementace appletu

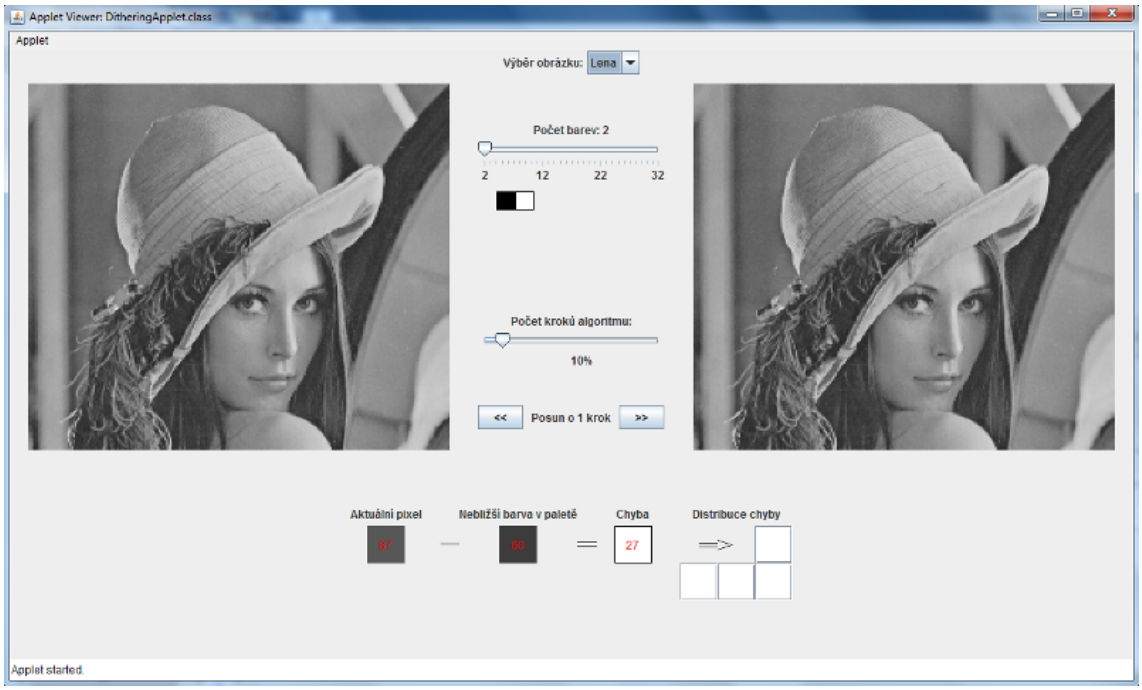
Hlavní třídou celého appletu je třída *KonvoluceApplet* ve které je definováno rozložení hlavních komponent a ovládá celý běh aplikace. Dalšími důležitými třídami jsou třída *Konvoluce*, ve které je implementován celý výpočet konvoluce. Na vstupu dostane vstupní obraz a filtr a vrátí nám výstupní obraz po konvoluci. Třída *Filter*, definuje jednotlivé filtry. Třída *KresliciPanel* načítá a vykresluje jednotlivé obrazy.

3.4 Applet dithering

Applet dithering ukazuje rozdělení chyby, která vznikne při redukci palety barev. V této implementaci je použita distribuce chyby na následující pixely podle algoritmu Floyd-Steinberg. Hlavními prvky rozhraní jsou vstupní a výstupní obraz. Mezi nimi se nachází ovládací panel, pomocí kterého se nastavují různé parametry aplikace. V dolní části je pak celá distribuce chyby názorně vyobrazena. Rozložení uživatelského rozhraní appletu je ukázáno na obr. 3.4.

Prostřednictvím ovládacího panelu je možné nastavit několik parametrů. Jako první je úplně nahoře možnost výběru vstupního obrazu. Následuje výběr počtu barev v paletě, na které se bude vstupní obraz redukovat. Pomocí posuvníku je možnost vybrat od 2 do 32 barev. Jednotlivé barvy v paletě se pak zobrazí pod posuvníkem. Dalším parametrem je počet procent hotových pixelů, tzn. pixely, které již byly redukovány a byla na ně aplikována metoda Floyd-Steinberg. Tato hodnota bude po spuštění aplikace nastavena na 10 % a výstupní obraz bude podle této hodnoty částečně zpracován. Dále pak bude mít uživatel možnost buď pomocí posuvníku tuto hodnotu měnit od 0 % do 100 % nebo postupovat po jednotlivých pixelech dopředu i zpět pomocí tlačítek v dolní části středního panelu a sledovat jak se mění výstupní obraz.

Ve spodním panelu jsou pak zobrazeny hodnoty podle aktuálně zpracovávaného pixelu. Nejprve je zobrazena barva a hodnota aktuálního pixelu, dále pak jeho nejbližší barva v paletě a následuje chyba, která je výsledkem rozdílu předešlých dvou hodnot. Jako poslední je ukázáno jak se chyba distribuuje na následující pixely



Obr. 3.4: Uživatelské rozhraní Appletu dithering.

v okolí aktuálního pixelu. Úplně nalevo spodního panelu se také nachází souřadnice zpracovávaného pixelu.

3.4.1 Funkčnost appletu

Po spuštění appletu se načte výchozí obraz a na 10 % jeho části je proveden dithering, při redukci barevné palety na 2 barvy. Výsledný obraz je pak zobrazen na pravé straně. Aktuálně zpracovávaný pixel je označen v originálním obraze červenou barvou a jeho souřadnice se zobrazí ve spodním panelu nalevo. Okamžitě je také zobrazena distribuce chyby aktuálního pixelu.

Uživatel má pak možnost měnit jednotlivé parametry. V rolovacím menu nahoře může zvolit výchozí obraz, posuvníkem změnit počet barev v paletě, dalším posuvníkem velikost části obrazu, která má být zpracována a nebo může sledovat postup ditheringu po jednotlivých pixelech a to dopředu i zpět pomocí šipek. Vždy okamžitě po změně jakéhokoliv z výše uvedených parametrů je hned proveden dithering, zobrazen výsledný obraz a je znázorněno rozprostření chyby posledního zpracovaného pixelu.

3.4.2 Implementace appletu

Hlavní třídou celého appletu je třída *DitheringApplet* ve které je definováno rozložení hlavních komponent a ovládá celý běh aplikace. Dalšími důležitými třídami jsou třída *Dithering*, ve které jsou implementovány všechny funkce pro výpočet ditheringu, především funkce *getDithering*, která pracuje s aktuálně nastaveným počtem barev a počtem zpracovaných procent a vrací výstupní obraz po ditheringu. Třída *SetPanel* definuje a řídí střední ovládací panel, pomocí kterého se mění jednotlivé parametry aplikace. Třída *KresliciPanel* načítá a vykresluje jednotlivé obrazy. A třída *DistribucePanel*, která definuje spodní panel appletu, ve kterém je detailně ukázána distribuce chyby aktuálního pixelu.

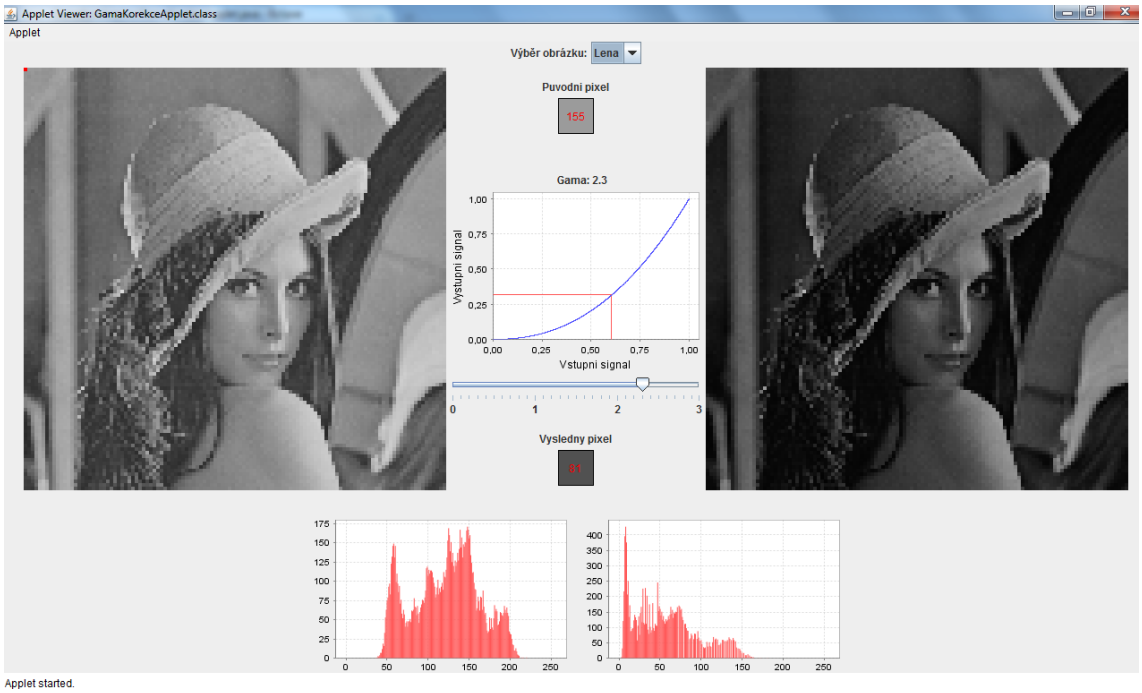
3.5 Applet gama korekce

Na vstupní obraz vlevo bude aplikována metoda gama korekce podle zadané hodnoty gama, která bude vybrána uživatelem pomocí posuvníku ve středním panelu pod grafem gama korekce. Výstupní obraz bude vykreslen vpravo. Ve středním panelu nahoře je zobrazena barva a hodnota aktuálního pixelu vstupního obrazu, který uživatel vybere kliknutím myši. Uprostřed panelu je zobrazená křivka podle které se gama korekce provádí. V grafu bude také vyznačen převod hodnoty vstupního pixelu na hodnotu výstupního pixelu. Pod grafem je dále zobrazena barva a hodnota výstupního pixelu. V dolním panelu je pod každým obrazem zobrazen histogram každého z nich a mezi nimi jsou souřadnice aktuálního pixelu. Rozvržení uživatelského rozhraní je ukázáno na obr. 3.5.

3.5.1 Funkčnost appletu

Ve chvíli, když se applet spustí, je načten výchozí vstupní obraz, na který je aplikována metoda gama korekce. Hodnota gama je zpočátku nastavená na hodnotu 2,3 a výsledný obraz je zobrazen na pravé straně. Jako aktuální pixel je vybrán pixel o souřadnicích $x = 0$, $y = 0$ a je označen ve vstupním obraze červenou barvou. Ve středním panelu jsou zobrazena hodnota a barva tohoto pixelu, hodnota a barva nového pixelu a křivka gama korekce s vyznačenou hodnotou aktuálního pixelu. Okamžitě po spuštění jsou také zobrazeny histogramy obou obrazů.

Uživatel má poté možnost zkoumat všechny pixely vstupního obrazu a to kliknutím na požadovaný pixel myší. Poté se pixel zbarví červeně, zobrazí se jeho souřadnice a všechny hodnoty s ním spojené. Dále má také možnost v horním panelu vybrat jiný obraz. Ihned po výběru se gama korekce znovu provede. Uživatel má možnost v průběhu měnit hodnotu gama pomocí posuvníku pod křivkou gama korekce. Ihned



Obr. 3.5: Uživatelské rozhraní Appletu gama korekce.

po změně parametru gama je provedena gama korekce na aktuálně vybraný obraz, změní se křivka, barva výstupního pixelu, histogram výstupního obrazu a výstupní obraz se zobrazí na pravé straně.

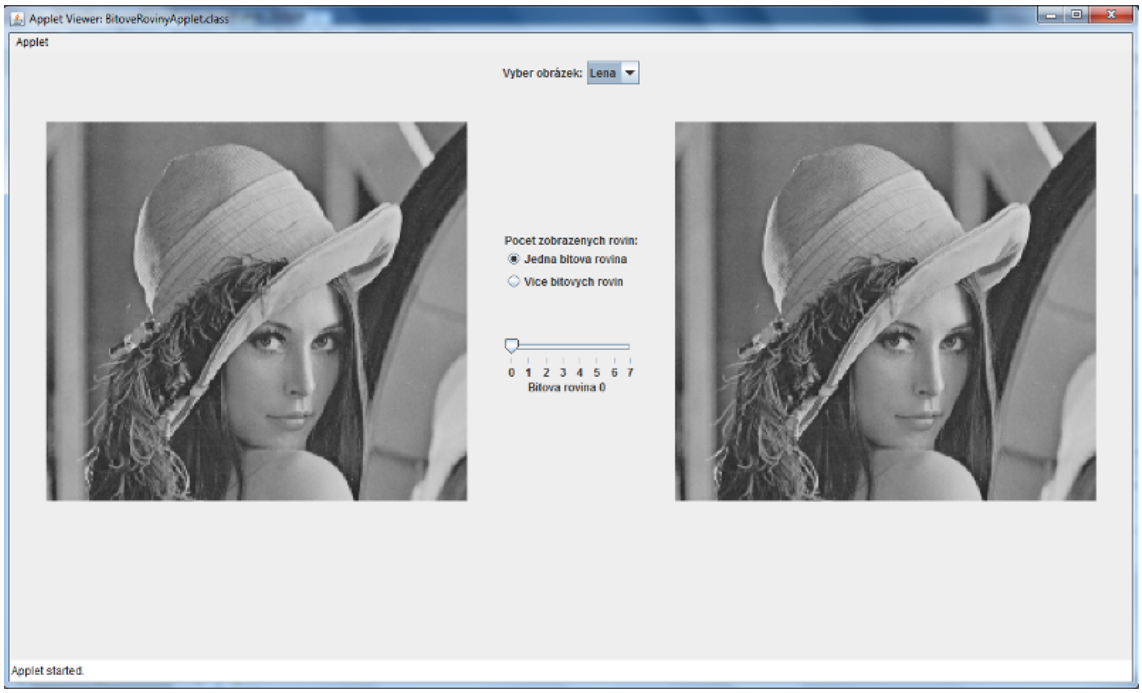
3.5.2 Implementace appletu

Hlavní třídou celého appletu je třída *GamaKorekceApplet* ve které je definováno rozložení hlavních komponent a ovládá celý běh aplikace. Dalšími důležitými třídami jsou třída *GamaKorekce*, která počítá celou metodu gama korekce a vrací výstupní obraz. Třída *StredniPanel* definuje a řídí střední panel, pomocí kterého se mění parametr gama a jsou zde zobrazeny hodnoty a barvy vstupního a výstupního pixelu. Třída *KresliciPanel* načítá a vykresluje jednotlivé obrazy. A třída *HistogramPanel*, implementuje histogramy vstupního a výstupního obrazu.

3.6 Applet bitové roviny

Tento applet zobrazuje jednotlivé bitové roviny nebo několik vybraných bitových rovin společně. Jelikož pracujeme s obrázky ve stupních šedi, je možné zobrazit až 8 bitových rovin. Rozvržení uživatelského rozhraní appletu je na obr. 3.6.

V levé části je vykreslen vstupní obraz a v pravé části jsou vykreslovány jednotlivé bitové roviny. V horní části appletu může uživatel vybrat vstupní obraz. Ve středním



Obr. 3.6: Uživatelské rozhraní Appletu bitové roviny.

panelu je výběr dvou možností zobrazení bitových rovin. Jedna možnost je zobrazení jedné bytové roviny a druhá zobrazení několika bitových rovin dohromady. Číslo požadované bitové roviny nebo více rovin je vybíráno posuvníkem.

3.6.1 Funkčnost appletu

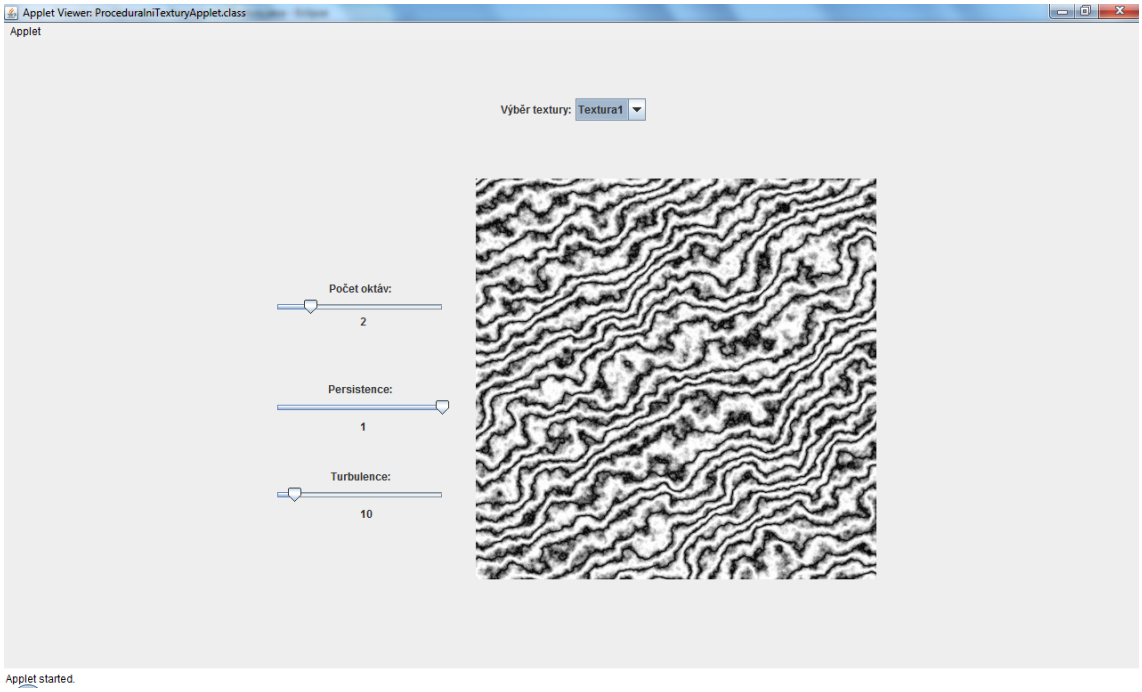
Po spuštění appletu je načten výchozí vstupní obraz a okamžitě je zobrazena jeho bitová rovina číslo 7, což je rovina s bitem, který nese největší informaci o pixelu. Ve středním panelu je pak možné nastavit volbu, kterou chce uživatel vidět. Při označení možnosti jedna bitová rovina se na výstupním obraze zobrazí právě jedna bitová rovina, jejíž číslo je vybráno pomocí posuvníku od 0 do 7. Při označení možnosti více bitových rovin se na výstupním obraze zobrazí několik bitových rovin dohromady. Uživatel vybere bitové roviny znovu tím samým posuvníkem, tentokrát se však vždy zobrazí roviny od 0 až po číslo které uživatel vybral. Tedy když uživatel nastaví posuvník na hodnotu 3, tak se na výstupním obraze zobrazí dohromady roviny 0, 1, 2, 3. V rolovací nabídce na horní straně appletu je možné také vybrat vstupní obraz, ihned po výběru se zobrazí bitová rovina, která je aktuálně nastavená posuvníkem.

3.6.2 Implementace appletu

Hlavní třídou celého appletu je třída *BitoveRovinyApplet* ve které je definováno rozložení hlavních komponent a ovládá celý běh aplikace. Dalšími důležitými třídami jsou třída *BitoveRoviny*, ve které je implementován celý výpočet a zobrazení jednotlivých bitových rovin. Třída *StredniPanel* definuje a řídí střední panel, pomocí kterého se mění parametry programu. Třída *KresliciPanel* načítá a vykresluje jednotlivé obrazy.

3.7 Applet procedurální textury

Applet zobrazuje vygenerovanou texturu a její změny v závislosti na nastavených parametrech. Uživatel si může zvolit z nabídky základních textur v horní polovině okna. Textura se okamžitě zobrazí ve vykreslovacím panelu. Nalevo od zobrazené textury jsou tři posuvníky pro nastavení jednotlivých parametrů textury. Rozvržení uživatelského rozhraní appletu je na obr. 3.7.



Obr. 3.7: Uživatelské rozhraní Appletu procedurální textury.

3.7.1 Funkčnost appletu

Po spuštění appletu je zobrazena textura, která má nastaveny parametry tak, aby bylo co nejvíce podobná mramoru. Uživatel má pak možnost nastavit počet oktáv,

které udávají detaily textury, tzn. čím více oktáv, tím bude struktura detailnější. Dále parametr persistence, který udává rychlost klesání vlivu oktáv na výslednou texturu. Třetím parametrem je turbulence, která deformuje původní zcela pravidelnou texturu. Ihned po změně parametru se textura změní.

3.7.2 Implementace appletu

Hlavní třídou celého appletu je třída *ProceduralniTexturyApplet* ve které je definováno rozložení hlavních komponent a ovládá celý běh aplikace. Dalšími důležitými třídami jsou třída *ProceduralniTextury*, ve které je implementován celý výpočet a zobrazení jednotlivých textur. Třída *SetPanel* definuje a řídí panel s posuvníky na změnu parametrů. Třída *KresliciPanel* načítá a vykresluje jednotlivé textury.

4 ZÁVĚR

Cílem této práce bylo seznámit se a implementovat určité metody zpracování obrazu ve formě Java appletů, které umožňují snadné a rychlé spuštění přes webové rozhraní a podpořit tak výuku v předmětech zabývajících se počítačovou grafikou. Tyto applety by měli nejen ukazovat konečný výsledek operace, ale především názorně ukázat jak fungují jednotlivé kroky daných operací.

V teoretické části práce jsou vysvětleny základní pojmy jako reprezentace obrazu v počítačové grafice, barevné modely a především jsou zde vysvětleny operace konvoluce, dithering, gama korekce, bitové roviny a procedurální textury, kterými se tato práce zabývá.

V praktické části projektu jsou kompletně popsány jednotlivé applety, jejich funkčnost a ukázka navrženého uživatelského rozhraní. U appletu konvoluce je použito několik základních používaných filtrů, jako jsou dolní propust, horní propust, jednotková matice, detekce hran a Gaussovo rozostření. Applet dithering je zaměřen na rozprostření chyby pomocí metody Floid-Steinberg. Applet bitové roviny ukazuje rozložení na jednotlivé roviny nebo součet několika bitových rovin. Gama korekce zobrazuje změnu jasu obrazu na základě hodnoty gama. A applet procedurální textury je zaměřen na generování textury mramoru.

Všechny cíle této práce se podařilo úspěšně splnit. Jednotlivé applety jsou kompletně implementovány a odzkoušeny, jak jejich uživatelské rozhraní, tak i všechny požadované funkce.

LITERATURA

[1] DANNHOFEROVÁ, J. *Počítačová grafika I* [online]. Mendelova univerzita v Brně, Brno: [cit. 28. 11. 2014]. Dostupné z URL: <https://is.mendelu.cz/eknihovna/opory/index.pl?opora=5>.

[2] ŽÁRA, J., BENEŠ, B., SOCHOR, J., FELKEL, P. *Moderní počítačová grafika*. Vyd. 2. Brno: Computer Press, 2004, 609 s. ISBN 80-251-0454-0.

[3] NETOPIL, V. *RGB a CMYK* [online]. Evropský polytechnický institut, Praha: [cit. 29. 11. 2014]. Dostupné z URL: <http://int1.webnode.cz/rgb/>.

[4] DOBEŠ, M. *Zpracování obrazu a algoritmy v C#*. 1. vyd. Praha: BEN – technická literatura, 2008, 143 s. ISBN 978-80-7300-233-6.

[5] JANOUCH, M. *Zpracování obrazu jednočipovým mikroprocesorem, Diplomová práce*. Praha: ČVUT Fakulta elektrotechnická, 2008, 72 s.

[6] JANČÍK, Z. *Vyukový program pro demonstraci principu barev a barevných modelů* [online]. Vysoké učení technické v Brně, Brno: [cit. 30. 11. 2014]. Dostupné z URL: <http://janosik.zlutaponorka.com/www-bp/node10.html>.

[7] HLAVÁČ, V; SEDLÁČEK, M. *Zpracování signálu a obrazu*. Praha: ČVUT Elektronická fakulta, 1999, 110 s.

[8] Konvoluce. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 25. 11. 2014]. Dostupné z URL: <http://cs.wikipedia.org/wiki/Konvoluce>.

[9] Dostupné z URL: <http://www.osel.cz/_img/img1267312240.jpg>.

[10] ŘÍHA, K. *Pokročilé techniky zpracování obrazu*. Vyd. 1. Brno: Ústav telekomunikací FEKT VUT, 2012, 143 s. ISBN 978-80-214-4894-0

[11] Gladišová, I., Mihalík, J., Zavacký, J. *Bezstratová kompresia obrazu pomocou stavového binárneho aritmetického kódovania jeho bitových rovín*. Košice: TU Fakulta elektrotechniky a informatiky, 2006.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

CMYK barevný model – Cyan, Magenta, Yellow, Key

HSV barevný model – Hue, Saturation, Value

JAVA programovací jazyk

RGB barevný model – Red, Green, Blue

YCrCb barevný model – jas (Y), chrominanční složky (red, blue)

A PŘÍLOHA

Obsah CD

- bakalářská práce ve formátu .pdf
- zdrojové kódy aplikace
- testovací sady obrázků